Application des Algorithmes Génétiques aux Flowshop Hybride

A. Soukhal, W. Duque, N. Sbiai, P. Martineau
LI/E3I – Université de Tours
64, Av. Jean Portalis
37200 Tours
tél : (33) 02.47.36.14.14 Fax : (33) 02.47.36.14.22
E-mail : soukhal@e3i.univ-tours.fr

Résumé : *Les problèmes d'ordonnancement dans les Flowshop Hybride sont de plus en plus étudiés. Nous étudions leur résolution à l'aide des méthodes de voisinage. Cet article présente les principes fondamentaux des Algorithmes Génétiques. Les adapter aux problèmes d'ordonnancement de type Flowshop nécessite un certain réglage des différents paramètres. Nous proposons une étude comparative de ces réglages à l'algorithme de base dans le cadre d'une application au FH.*

Mots-clés : Ordonnancement, Flowshop Hybride, Méthodes de voisinage.

I Introduction

Les problèmes d'ordonnancement constituent une branche de la Recherche Opérationnelle. On rencontre des problèmes d'ordonnancement dans toutes nos activités quotidiennes : dans la gestion de notre emploi du temps, dans les ateliers de production, dans les systèmes d'exploitation, ... Pour aborder leur résolution, on identifie tout d'abord l'organisation physique du système où on trouve principalement du Flowshop, du Jobshop, ou de l'Openshop. Depuis quelques années, les chercheurs portent leur intérêt sur les Flowshop Hybrides (FH), NP-difficile dans le cas général [Gupta,88].

Nos travaux portent sur la résolution de ces problèmes par des algorithmes de voisinage (Recuit Simulé, Méthode Tabou, Algorithme Génétique, ...) [Soukhal et al.,99]. En particulier, cet article détaille l'adaptation des Algorithmes Génétiques (AG) au FH. En effet, les AG sont souvent critiqués parce qu'ils demandent une période de réglage longue et délicate. Nous travaillons sur la base de la forme basic de l'AG. Une étude complète sur l'application et l'adaptation des AG au FH plus efficacement est encours de réalisation [Soukhal et al.,99] qui porte sur le réglage des opérateurs génétiques (codage, mutation, croisement...).

Dans une première partie, nous rappellerons la structure des FH avant de décrire la forme générale d'un AG en insistant sur les points clés à régler. Ensuite, nous décrivons les alternatives auquel le développeur est confronté lors de l'application d'un AG au FH et nous indiquons les choix que nous avons effectués en les justifiant. Les résultats d'une étude comparative sont présentés pour valider ces réglages. L'article s'achève par une conclusion et quelques perspectives.

II Etat de l'Art

Cet état de l'Art porte sur le Flowshop Hybride puis sur les Algorithmes Génétiques car notre article se situe au croisement de ces deux thèmes.

II.1 Flowshop Hybride

On considère qu'on a affaire à un atelier organisé en FH lorsque tous les travaux doivent passer dans le même ordre sur L étages (ateliers), st_1, st_2,..., st_L où chaque étage regroupe une ou plusieurs machines parallèles, identiques ou non.

Classiquement, n tâches (T_1, T_2, ..., T_n) sont prêtes à s'exécuter à la date t=0. A chaque étage, une opération est traitée par une seule machine et, à un instant donné, une machine traite au plus une seule opération. Il n'y a pas de contraintes de précédence, pas de préemption. Les temps de transport d'un étage à un autre sont négligés ou inclus dans les durées opératoires. La durée d'exécution de la $j^{ième}$ opération O_{ij} de la tâche T_i est connue, fixe, indépendante et égale à P_{ij}.
Chaque étage dispose d'un stock d'entrée/sortie de capacité illimitée.
En tenant compte des hypothèses classiques en ordonnancement, ce problème est défini par la notation suivante : $FH,(PM_i)_{i=1}^{L} // C\,max$ [Vignier, 97].

La minimisation du Cmax pour les systèmes de type Flowshop Hybride est un problème NP-difficile au sens fort. En effet, dans [Gupta,88], Gupta a montré que lorsque l'on considère deux étages dont au moins un est constitué de plus d'une machine identique en parallèle le problème FH est NP-difficile au sens fort.

Les principaux travaux des chercheurs dans le cadre de cette problématique tournent autour:
1- du calcul des bornes, en particulier, les travaux de Santos et al. [Santos et al.,95] qui permettent de calculer la borne inférieure pour un problème où le nombre d'étages est quelconque.
2- de la résolution par méthodes exactes, à savoir, les méthodes par séparation et évaluation [Brah et al.,91], [Moursli,99] , la programmation linéaire en nombres entiers [Guinet et al.,96]. Ces deux méthodes permettent d'obtenir une solution optimale du problème, mais sont très limitées en taille.
3- de la résolution par méthodes approchées comme les algorithmes de liste, les algorithmes de voisinage... Plusieurs heuristiques ont été proposées, [Gupta,88], [Deal et al.,91], ... (voire [Vignier,97]). Elles permettent de résoudre des problèmes avec un grand nombre de travaux en un temps de calcul raisonnable.

Dans [GOTHA,93], les auteurs définissent cinq grandes familles de méthodes de résolution :
- Méthodes par construction progressive : ce sont des méthodes itératives guidées par des règles de priorité pour le choix des tâches à exécuter ;
- Méthodes par voisinages ;
- Méthodes par décomposition ;
- Méthodes par relaxation ;
- Méthodes liées à l'intelligences artificielle.

Chacune de ces familles peut regrouper à la fois des méthodes exactes et des méthodes approchées.

II.2 Algorithme génétique

Les algorithmes stochastiques ont montré leur capacité à traiter des problèmes à très forte combinatoire ou des problèmes dont les contraintes qui ne peuvent se modéliser simplement. Ce sont des méthodes d'amélioration itératives. Elles sont basées sur la notion de voisinage d'une solution admissible. Un voisin d'une configuration est une modification de celle-ci, par exemple dans un problème d'ordonnancement, l'obtention d'une séquence voisine peut s'effectuer par la permutation de deux ou plus de ses composantes.

L'algorithme de base est l'algorithme de descente stochastique qui accepte une solution voisine si elle est meilleure que (ou équivalente à) la précédente. Un inconvénient majeur de la descente stochastique est que l'algorithme s'arrête dès qu'un premier optimum local est atteint en partant d'une solution initiale choisie. En général, cette solution initiale est choisie aléatoirement parmi toutes les configurations de l'ensemble des configurations Ω du problème considéré.

Pour pallier quelques uns de ces inconvénients, on a imaginé un certain nombre d'approches :
- Exécution de l'algorithme pour un grand nombre de configurations initiales ;
- Utilisation des informations glanées lors de précédentes exécutions pour améliorer le choix de la configuration initiale ;
- Introduction d'une fonction de voisinage plus complexe afin d'élargir le voisinage ;
- Acceptation de configurations correspondant à une dégradation de la fonction de coût (pour permettre d'échapper à un minimum local);

A partir de là d'autres méthodes basées sur la notion de voisinage ont vu le jour telles que : le recuit simulé , la recherche tabou et les algorithmes génétiques. Ces trois méthodes sont fortement utilisées pour la résolution des problèmes de très forte combinatoire, notamment pour les problèmes d'ordonnancement. Signalons tout de même, que pour la résolution des problèmes de type Flowshop Hybride, elles sont très peu utilisées. [Vignier,97], [Djerid,96], [Portmann,96] ont proposé une méthode de résolution basée sur les algorithmes génétiques. Dans [Nowicki et al.,98], les auteurs utilisent la méthode tabou pour la résolution d'un problème de Flowshop à L étages. [Haouari et al.,97] étudient le problème $FH2,(PM_i)_{i=1}^{k} // C\,max$, problème à deux étages. Ils proposent une amélioration de la borne inférieure présentée par Lee et al. [Lee et al.,94], et ils proposent le recuit simulé comme méthode de résolution.

Les algorithmes génétiques ont été développés par Holland et son groupe de travail à l'Université de Michigan dans les années 70 [Holland,75]. Depuis plusieurs extensions de ces algorithmes connus sous le nom des algorithmes évolutionnistes [Goldberg,89], [Gen et al,97], [Caux et al.,95] sont utilisés pour la résolution des problèmes à forte combinatoire.

L'idée des algorithmes génétiques est de reproduire l'évolution naturelle d'organismes (individu) génération après génération, en respectant les phénomènes d'hérédité et la loi de survie énoncée par Darwin : les individus les mieux adaptés au milieu survivent et donnent une descendance.

Un individu est caractérisé par un ensemble de chromosomes issues de la *recombinaison* des chromosomes de ses deux parents par *croisement* (*cross-over*) ou/et *mutation* (*mutation*). Les parents sont considérés mieux adaptés au milieu et produisent un ou plusieurs individus fils (souvent dans la littérature deux fils, [Djerid et al.,96] développent une procédure de croisement qui génère quatre fils).

On utilise un vocabulaire similaire à celui de la génétique naturelle, on parlera de population d'individus où chaque individu représente une solution au problème à résoudre. Les individus sont des représentations ou des codages, sous forme de chaînes, de solutions admissibles. Tous les individus sont évalués à chaque génération et il leur est associé une force appelée en anglais (*fitness*), cette force peut être mesurée par la fonction objectif correspondante.

Les principales étapes d'un algorithme génétique (AG) sont :

1. Définir un codage du problème et créer une population initiale.
2. Appliquer aléatoirement les opérateurs de croisement et de mutation sur un certain nombre d'individus, ensuite évaluer la force de l'ensemble des individus pour pouvoir sélectionner ceux qu'on gardera dans la nouvelle population.
3. Aller à 2 si la condition d'arrêt n'est pas vérifiée.

La mise en œuvre d'un AG nécessite donc de régler certains paramètres :

- Définir un critère d'arrêt comme une valeur seuil ou un nombre maximum d'itérations.
- Définir les probabilités de croisement et de mutations.
- Dimensionner la taille de la population générée par l'algorithme.

Très souvent, ces paramètres sont réglés de manière empirique. Or, ce réglage a un impact très important sur la convergence de l'algorithme et sur les résultats obtenus. On détaillera par la suite les différentes étapes de cet algorithme adaptées à la résolution de notre problème.

II.2.a Codage des individus

En langage génétique, une solution est un génome, il peut être composé de plusieurs chromosomes (par exemple, chaque chromosome représente l'ordonnancement et l'affectation des tâches dans chaque étage pour le problème FH). Un chromosome est composé de gènes. Un gène est une particularité ou propriété du problème considéré. Le codage de ces gènes s'effectue selon l'un des deux types :

1- Codage binaire en 0 et 1 ;
2- Codage réel.

II.2.b Population initiale

Contrairement aux autres méthodes de voisinage tels le recuit simulé et tabou, les AG demandent plus qu'une solution initiale un ensemble de solutions admissibles appelé *population initiale*. La construction de la population initiale s'effectue soit [Caux et al.,95]:

1- De manière aléatoire : cette méthode permet d'avoir une population variée qui donne la possibilité d'explorer les diverses zones de l'espace des solutions.
2- De manière heuristique : on initialise la population par des solutions provenant d'heuristiques dans l'espoir de les voir s'améliorer lors des générations successives. Mais l'inconvénient est que cette méthode peut influer sur l'AG et peut le faire converger trop rapidement vers un optimum local.
3- Par combinaison des deux méthodes : pour avoir une population variée, on incorpore des solutions provenant d'heuristiques à une population de solutions aléatoires.
4- Par duplication et évolution : dans le cas où le problème est fortement contraint, trouver un ensemble de solutions admissibles n'est pas une chose facile. Pour cela, il suffit de trouver, au moins, une seule solution admissible puis d'appliquer des fonctions de croisement et de mutation sur cette solution autant de fois que la taille de la population considérée. Ainsi on obtient une population initiale qui nous permettra de lancer l'AG.

II.2.c La sélection

La sélection s'effectue selon l'une des deux méthodes : déterministe ou mixte (hasard + déterministe) [Gen et al.,97].

La sélection déterministe :

- Elitiste : prendre le meilleur individu selon la fonction objective.
- Troncature : sélectionner le meilleur et le dupliquer Q fois (Q inférieur à la taille de la population choisie par l'utilisateur).
- Sélection par bloc : prendre un nombre Q de copies des meilleurs chromosomes de population.
- Remplacement de génération : on remplace les n mauvais parents par leurs fils obtenues lors de mutations et de croisements.

La sélection mixte : comme son nom l'indique, combine le « hasard » et le déterministe :

- Sélection tournante : choisir aléatoirement un ensemble de chromosomes et prendre le meilleur de l'ensemble pour la reproduction. La taille de l'ensemble choisi est la taille tournante ≥ 2.
- Sélection tournante stochastique : connue sous le nom de roulette [Gold,89]. C'est la plus classique. Elle consiste à associer à chaque individu x_i une probabilité proportionnelle d'être sélectionné:

$$p(x_i) = \frac{C(x_i)}{\sum_{k=1}^{pop\_size} C(x_k)}$$

puis la sélection des individus s'effectue de la façon suivante :
Calculer la probabilité d'apparition cumulée d'un individu x_i

$$q_i = \sum_{j=1}^{i} P(x_j)$$

1- Générer un nombre aléatoire r compris entre 0 et 1.
2- Si $r<q_1$, sélectionner alors le premier individu x_1 ; sinon sélectionner l'individu x_i
3- $(2 \le i \le N)$ tel que $q_{i-1} < r \le q_i$.
4- Répété pop_size fois (où pop_size est le nombre d'individus dans la population).

Avec ce principe, un individu fort est sélectionné plusieurs fois. Au contraire, un individu faible a moins de chance d'être sélectionné.

- Méthode stochastique universelle simple : dupliquer les individus selon leur probabilité proportionnelle et ensuite appliquer la méthode élitiste pour la sélection.

Signalons tout de même qu'il y a plusieurs autres façons de calculer cette probabilité par exemple [Gen et al.,97] : $P(x_i) = 2*r / (q(q+1))$ où r et q sont respectivement l'indice du $r^{ème}$ chromosome et l'indice du meilleur individu.

II.2.d Opérateurs génétiques

Les opérateurs génétiques permettent de générer une nouvelle population d'individu à partir de la population courante. Il existe deux grandes catégories d'opérateurs génétiques : les mutations et les croisements.

Opérateur de croisement

L'opérateur de croisement a pour but d'essayer de créer de « bons » individus en agissant sur la structure des chromosomes des parents. Il existe de nombreux types de croisement dont:

1- Le croisement à un point, inspiré du processus biologique, coupe chacun des deux parents sélectionnés en un même point choisi aléatoirement et produit deux enfants en échangeant les parties coupée.

Position de coupure

Parent1

a_1	a_2	...	a_{pos}	a_{pos+1}	...	a_n

Parent2

b_1	b_2	...	b_{pos}	b_{pos+1}	....	b_n

Position de coupure

fils1

a_1	a_2	...	a_{pos}	b_{pos+1}	....	b_n

fils2

b_1	b_2	...	b_{pos}	a_{pos+1}	....	a_n

Ce processus est appliqué à chaque paire de chromosomes sélectionnés avec une certaine probabilité p_{cross}. Les paires sont recopiées sans modification dans la génération suivante avec la probabilité 1-p_{cross}.

2- Le croisement multi-points dit k-points k=2..K. Il est similaire au croisement à un point sauf qu'il coupe chacun des deux parents en k points choisis aléatoirement. Ce croisement produit deux descendants. On recopie les mêmes gènes du parent1 jusqu'au point de croisement ou recopie les gènes du parent2 ainsi de suite. [Djerid,97] a défini une méthode de croisement à k-points qui nous permet de générer quatre fils au plus (quatre fils si leur chromosomes sont différents).
Il existe d'autres types de croisements : croisement uniforme de JONG [GOLD,89] ; le croisement uniforme de Syswerda (voire [Davis,91]).

Opérateur de mutation

L'opérateur de mutation a pour but de diversifier les individus dans la population. La mutation la plus classique consiste à sélectionner aléatoirement, avec une probabilité p_{mut}, un gène du chromosome d'un individu et à modifier sa valeur, ou faire un échange de deux gènes dans un chromosome. Il existe d'autre types de mutation comme inverser l'ordre des gènes entre deux coupes [Gen et al.,97]...
Nous présentons la structure générale de l'algorithme génétique :

```
Procedure AG
Debut
    Mettre en forme les données.
    t ← 0.
    Rechercher une population initiale P(0).
    Evaluer les individus de la population P(0) et noter la valeur du meilleur individu.
    Tantque (la condition d'arrêt n'est pas satisfaite)
        Faire
            Evaluer l'ensemble des individus P(t)
            Sélectionner des futurs parents dans P(t).
            Appliquer le croisement avec une probabilité p_cross .
            Appliquer la mutation avec une probabilité p_mut.
            t ← t+1
        Fait
    FinTantque
Fin
```

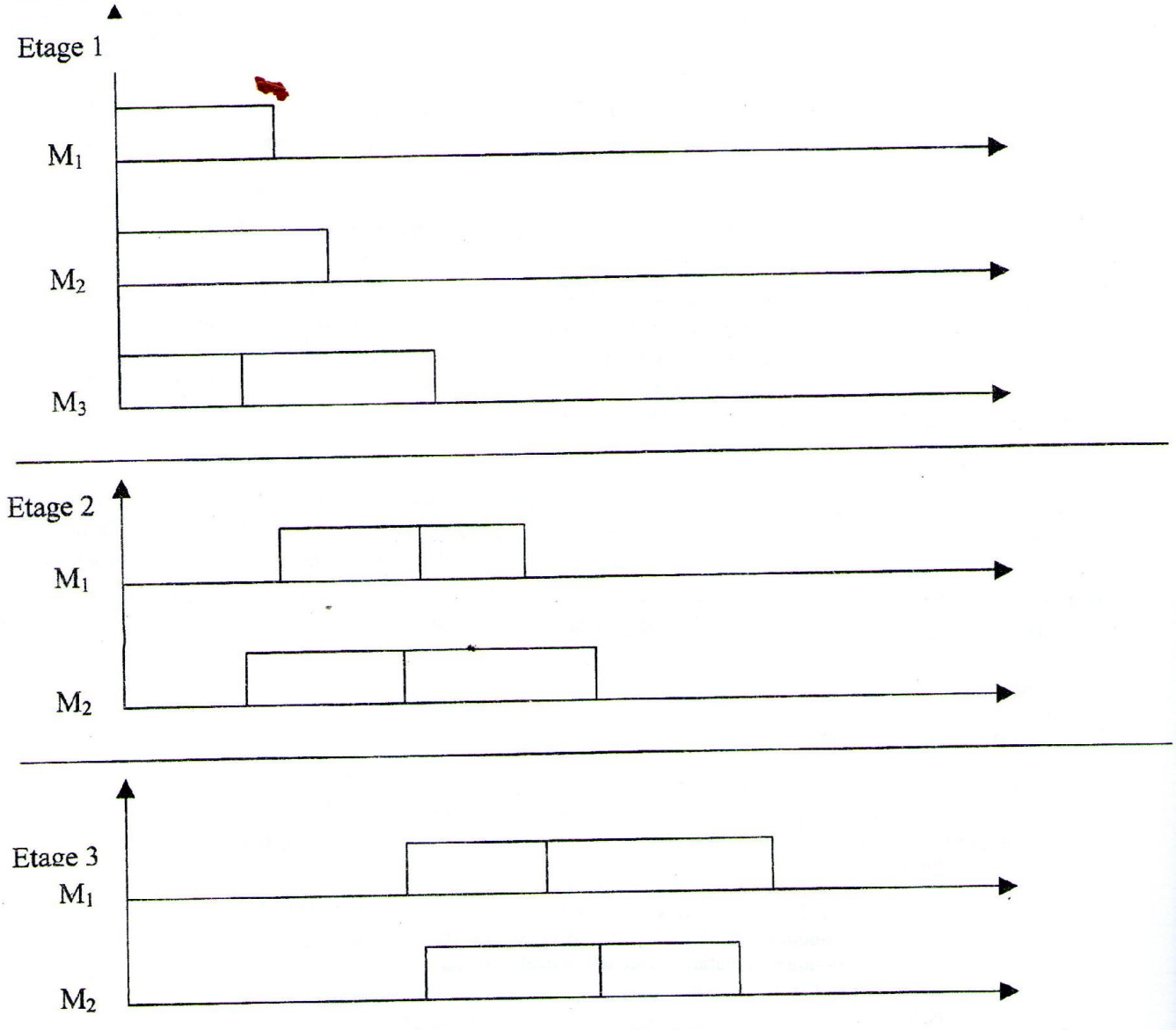
III Présentation de l'AG

Pour l'application de cet algorithme à notre problème, on présentera les choix utilisés pour définir le codage des données, l'initialisation de la population, l'évaluation, les opérateurs génétiques (opérateur de croisement et de mutation) et la procédure de sélection.
Nous travaillons sur la base de la forme basic de l'AG dans le but de régler au mieux ses paramètres, à savoir, la probabilité de croisement, la probabilité de mutation et la taille de la population initiale. Appliquer plus efficacement l'AG au FH nécessite une étude plus complète sur le choix des différents opérateurs génétiques, le codage... Cette étude est encours de réalisation.

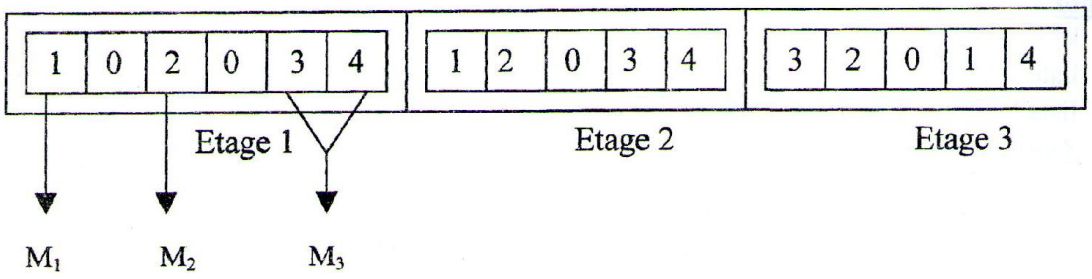
III.1 *Codage des données*

La population est l'ensemble des individus qui représentent l'ordre de passage des opérations dans chaque étage. L'individu sera représenté par un seul chromosome. Celui-ci est codé sous la forme d'une chaîne de caractères, cette chaîne est constituée des numéros des travaux dans l'ordre de leur ordonnancement. Le passage d'une machine à une autre est traduit par un zéro (un gène =0 si et seulement si on change de machine).

Exemple
Soit un Flowshop Hybride à 3 étages et 4 travaux, les 3 étages contiennent 3, 2, 2 machines successivement.



Le codage de cette séquence (chromosome) est le suivant :



III.2 Construction de la population initiale

Comme on l'a vu, il existe plusieurs façons pour construire la population initiale. Notre choix a été la combinaison entre le tirage aléatoire et l'utilisation d'une heuristique H (présentée ci-dessous).

- Le tirage pseudo-aléatoire: on affecte les tâches au hasard aux machines disponibles en tenant compte des contraintes de précédence entre les étages, les machines doivent être toutes utilisées ;
- L'heuristique H : cette heuristique fournit un ordonnancement pour le problème, noté:

$$\text{Pb, } FH,(PM_i)_{i=1}^{L} // C\,max\,.$$

H se déroule en deux phases :

Phase1 : consiste à résoudre le problème à deux étages, noté Pb', fictifs (L-1) fois. Le premier étage est l'ensemble des k premiers étages du problème primale Pb où k varie de 1 à (L-1). Le deuxième étage est défini par les (L-k) derniers étages.

A l'étape k, on calcule d'abord les durées opératoires de chaque tâche au premier et au deuxième étage (P'_{i1} et P'_{i2}) de Pb' :

$$P'_{i1} = \sum_{j=1}^{k} P_{ij}\,, \qquad P^{i}_{i2} = \sum_{j=k+1}^{L} P_{ij}$$

L'affectation et l'ordonnancement des tâches sur les machines s'effectue selon le principe suivant : dès qu'une tâche est prête à s'exécuter, alors elle est affectée à la machine libre au plus tôt. En cas de conflit entre les tâches (plusieurs tâches disponibles au même moment) on se référera à l'ordre de Johnson [Johnson,54]. En 1954, Johnson donne un algorithme optimal pour la résolution du problème du type Flowshop classique à deux machines $F2//C\,max$ qui consiste à libérer la première machine au plutôt tout en minimisant les temps morts sur la deuxième machine [Johnson,54].

Pour chaque valeur k (k=1..L) on recalcule le $Cmax^k$ ainsi que sa séquence correspondante S_k.

On ne retient que la séquence S' qui donne le $\min_{k=1}^{L}(C\,max^{k})$

Phase2 : consiste à résoudre le problème Pb par l'affectation des tâches prêtes à s'exécuter aux machines disponibles au plutôt. En cas de conflit entre les tâches(comme le cas au premier étage), on ordonnance selon l'ordre S'.

L'algorithme H :

Soit k un entier, L le nombre d'étage, S_k et $Cmax^k$ l'ordonnancement et le Cmax obtenus à l'étape k:

Debut
k=1 ;
liste1=∅ ;
liste2=∅ ;
Tantque (k ≤ L)
Faire
Pour toute tâche T_i (i= 1..n):

Calculer $P'_{i1} = \sum_{j=1}^{k} P_{ij}$; $P^i_{i2} = \sum_{j=k+1}^{L} P_{ij}$

Si $P'_{i1} \leq P'_{i2}$ alors mettre T_i dans liste1 ;
Sinon mettre T_i dans liste2 ;

FinPour
Ordonnancer les tâches de la liste1 selon l'ordre croissant des P'_{i1} puis les tâches de la liste2 selon l'ordre décroissant des P'_{i2} suivant la machine libre au plutôt. Soit S_k l'ordre obtenu.
Calculer et retenir $Cmax^k$ et la séquence obtenue S_k.
k=k+1 ;
Fait
FinTantque
Cmax'= $\min_{k=1}^{L}(Cmax^k)$, S' séquence correspondante.
Résoudre le problème Pb en affectant les tâches prêtes à s'exécuter aux machines disponibles au plus tôt. En cas de conflit, l'ordre des tâches est déterminé par S'
Fin

III.3 Evaluation

Le problème que l'on traite est un problème de minimisation (minimisation de Cmax). L'évaluation permet de mesurer la force de chaque individu. Cette fonction prend la forme suivante : eval(S_k) = Cmax où S_k est le chromosome k.

Plus le C_{max} est grand, plus l'évaluation eval(S_k) est grande, donc moins cet individu aura de chance d'être choisi dans la prochaine itération (ici le plus fort est celui qui a le plus petit makespan).

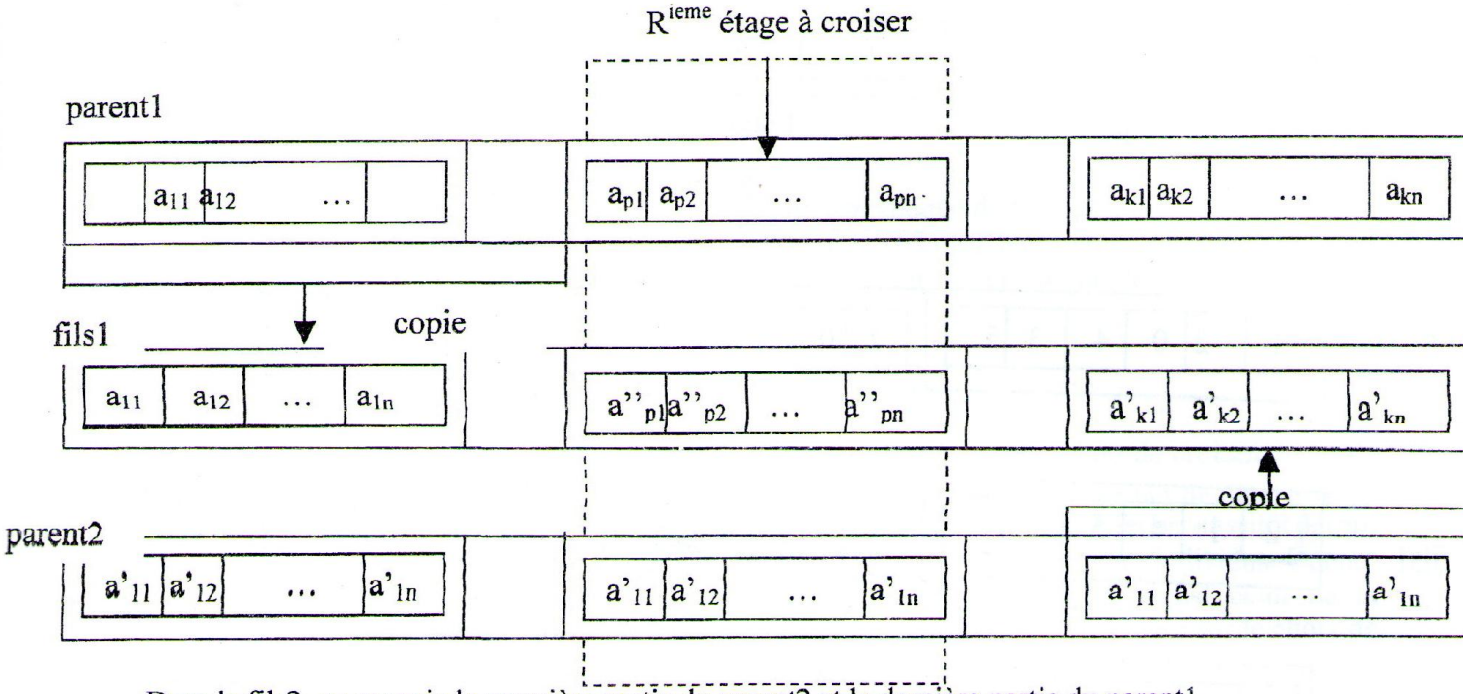
III.4 Opérateur génétique

A partir de la population courante, on utilise deux opérateurs génétiques pour générer les individus de la nouvelle population.
Les individus parents qu'on veut croiser ou muter pour créer les nouveaux fils sont choisis à partir de l'ancienne population avec une probabilité p_{cross} pour le croisement et p_{mut} pour la mutation.

III.4.a Croisement

Le croisement considéré est le croisement à un point. Comme on l'avait vu dans le codage des données, on représente chaque individu, solution de notre problème sous forme d'un tableau ; chacune de ses cases correspond à un ordonnancement sur un étage.

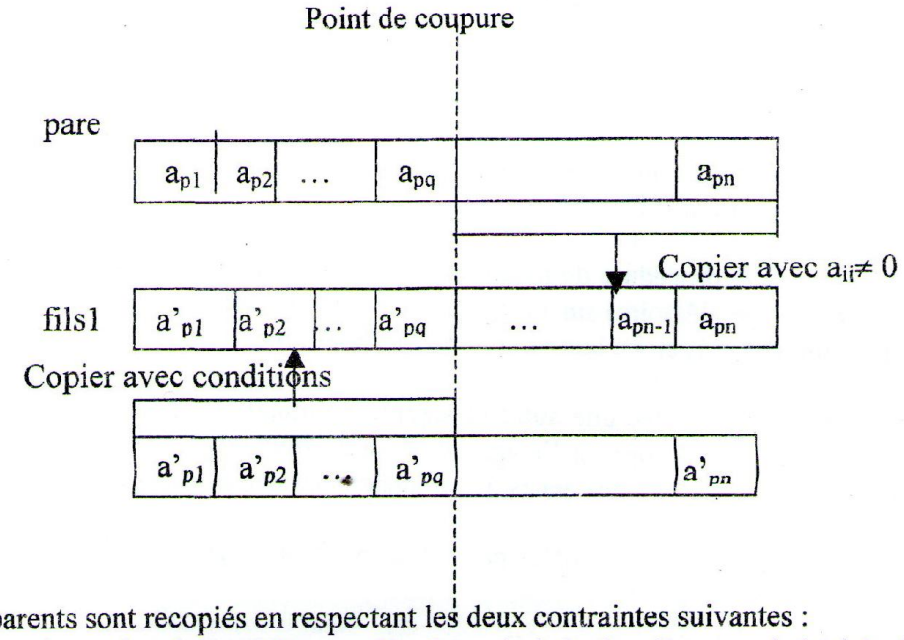
On sélectionne deux parents parmi les individus de la population en utilisant la procédure de sélection des parents et avec une probabilité P_{cross} variant entre 25% et 75%. La technique consiste à choisir un étage R aléatoirement sur lequel on va effectuer le croisement pour créer l'ordonnancement et l'affectation des tâches au $R^{ième}$ étage dans les deux fils. Les autres étages sont recopiés des parents.



Dans le fils2, on recopie la première partie du parent2 et la dernière partie du parent1.

Le croisement des individus au R ième étage s'effectue comme suit :

les machines de l'étage p du parent1 sont héritées par le fils1 (c-à-d. les zéros représentant les machines) tandis que celles du parent2 sont héritées par le fils2. On choisit un point de coupure aléatoirement, la deuxième partie du parent1 est recopiée dans fils1 (les tâches mais pas les machines) et la première partie est prise du parent2 tout en vérifiant que les tâches n'étaient pas déjà affectées lors de la copie de la première partie.



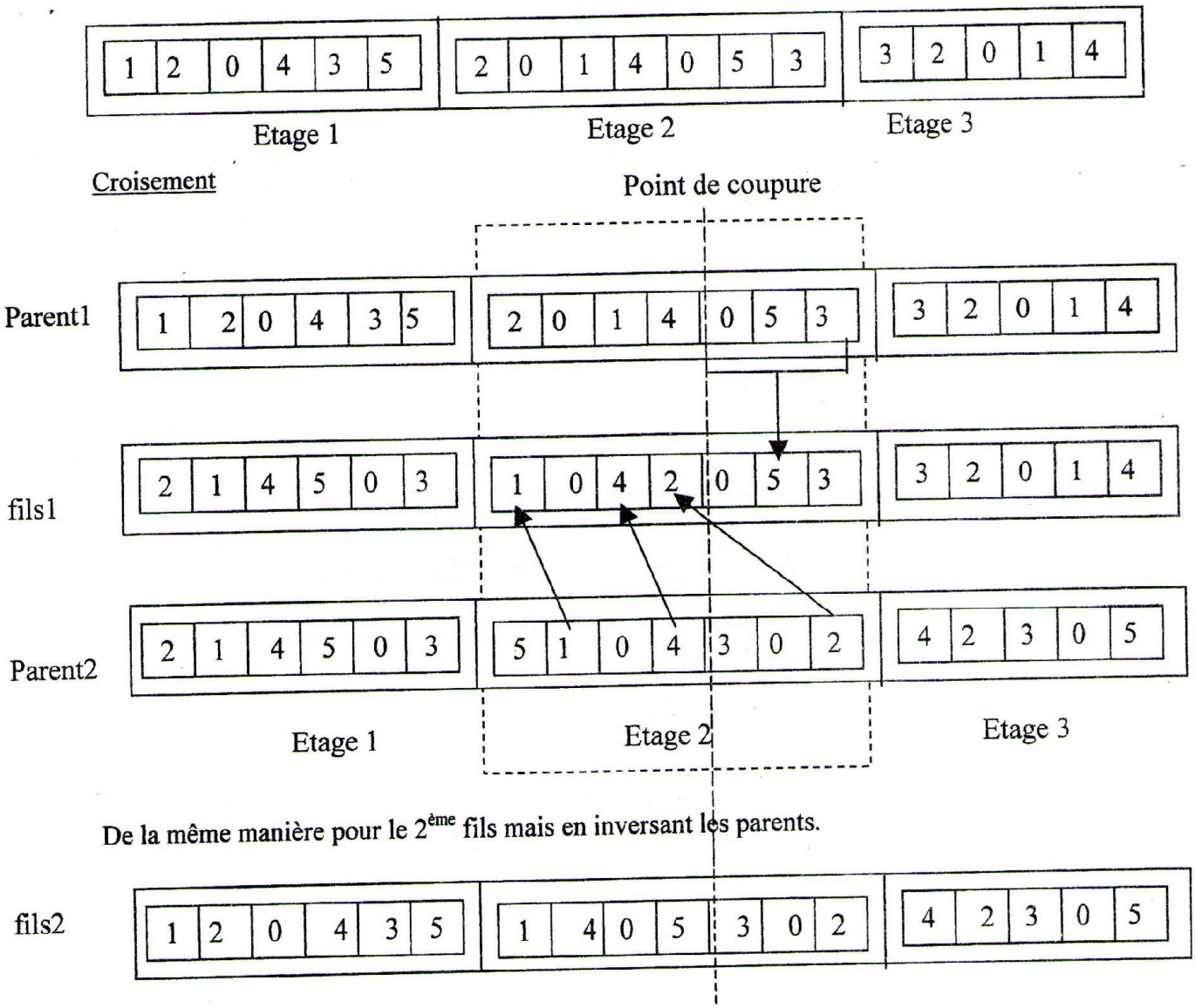
Les gènes des parents sont recopiés en respectant les deux contraintes suivantes :

- On ne recopie pas les tâches déjà recopiées à partir de la deuxième partie héritée du parent1 ;
- On recopie seulement les gènes différents de zéro ($a_{ij} \neq 0$).

Exemple :

Reprenons l'exemple vu précédemment pour le FH à 3 étages et 5 tâches, les étages contiennent successivement 2, 3, 2 machines.

L'ordonnancement et l'affectation des tâches aux différentes machines sont décrits par le chromosome suivant :



De la même manière pour le 2ème fils mais en inversant les parents.

III.4.b Mutation

Les individus à muter sont sélectionnés avec une probabilité p_{mut}. [Gen et al,97] indique que la valeur de cette probabilité doit varier entre 1% et 15%.

Nous proposons une nouvelle procédure de mutation par insertion de gène qui consiste à choisir deux positions pos1 et pos2 (tirage aléatoire) sur l'étage où on veut appliquer la mutation, puis à insérer le gène de pos2 en position pos1+1, en ne laissant pas de machine libre.

Cette procédure a été comparée avec une autre procédure de mutation, transposition de gène. Cette procédure consiste à choisir aléatoirement l'étage où on veut effectuer la mutation. On échange alors deux gènes à cet étage. Dans le choix des gènes, il y a quelques transpositions à interdire :

- Echange de deux machines représentées par la transposition de gène à valeur 0 (ils représentent un changement de machine) ;
- Pas de machine oisive, ceci s'exprime en interdisant deux 0 successifs.

La mutation par insertion donne de bons résultats par rapport à la mutation par transposition de gènes [Soukhal et al.,99].

La Sélection :

Pour la sélection de la nouvelle population nous proposons la méthode **élitiste** présentée en détail au début de l'article. Le choix des deux parents pour le croisement s'effectue selon la méthode tournante stochastique.

Critère d'arrêt :
L'algorithme s'arrête après un nombre fixe de génération (100).

IV Expérimentations

Nous proposons d'évaluer le comportement de l'algorithme génétique de base pour la résolution de FH de manière à juger de son adéquation à la résolution de tels problèmes. Nous étudions successivement l'influence des paramètres P_{cross}, P_{mut} et pop_size en les faisant varier autour de valeurs standards (respectivement 0.75, 0.15 et 100).

IV.1 Réglage de l'AG

Le réglage des paramètres est un point délicat des AG. Le choix de la procédure de croisement et de mutation joue un rôle important sur la convergence des AG. La mutation a pour but de diversifier la recherche, or une valeur trop grande de probabilité Pmut a peu d'intérêt car son effet est aussitôt détruit par la sélection suivante qui favorise(avec une probabilité très forte) les meilleurs individus. Donner une probabilité trop faible entraîne une négligence du principe des AG qui est celui de la diversification de la recherche.(Cette dernière réflexion nous conduit à nous interroger sur le choix de la taille de la population.)

D'autre part, le croisement est l'opérateur de base de toute reproduction. Donner une probabilité trop faible empêche l'évolution et le développement de la population. L'augmenter (proche de 1) entraîne trop de changement dans la population et élimine les parents trop rapidement. Pour cela il y a deux approches:

- Effectuer la sélection sur l'ensemble de la population des parents et des enfants en même temps. A ce moment là, on augmente la probabilité de croisement...
- Effectuer la sélection sur l'ensemble constitué des fils (dus aux croisements et aux mutations) et des parents sélectionnés n'ayant pas participas au croisement et à la mutation pendant cette génération. Donc est il préférable de ne pas augmenter trop la probabilité afin de garder des ancêtres de génération en génération.

Dans la littérature, ces paramètres sont réglés empiriquement.

IV.2 Evaluation

Les campagnes de test effectuées ont pour objectif d'évaluer l'influence du choix de ces paramètres lors de l'application des AG au FH. Les paramètres étudiés sont la probabilité de croisement, la probabilité de mutation et la taille de la population. Pour chacun, nous avons effectué trois séries de jeux, correspondant à trois tailles d'atelier: 2étages, 6 étages et 10 étages. Chacun de ces jeux est constitué de 50 jeux de données, réparés en 5 ensembles de 10 jeux de données (correspondant à 5 nombres de travaux allant de 10 à 50).

IV.2.a En fonction de p_{cross}

La probabilité de croisement pcross est un élément prépondérant. Elle est classiquement réglée à 75%.

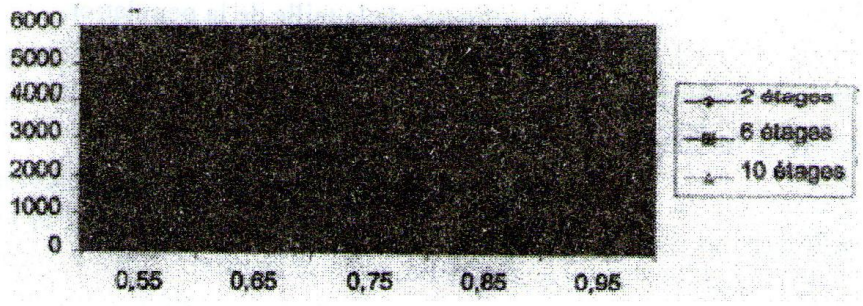


Figure 8 Influence de la probabilité de croisement.

Nous l'avons fait varier de 55 à 95 %. L'influence de ce paramètre sur la valeur du critère obtenue est notable. Sur les jeux d'essais à le étages, on obtient jusqu'à 7 % d'écart entre le meilleur et le pire des résultats. Les tests confirment que 75 % est une bonne valeur de réglage par défaut.

IV.2.b En fonction de P_{mut}

La probabilité de mutation doit assurer la diversification de la population. Or l'analyse fine de celle-ci montre qu'elle est déjà très diversifiée, voire trop. Il est donc difficile de conclure au vue de la courbe représentant le Cmax en fonction de P_{mut}

IV.2.c En fonction de la taille de la population

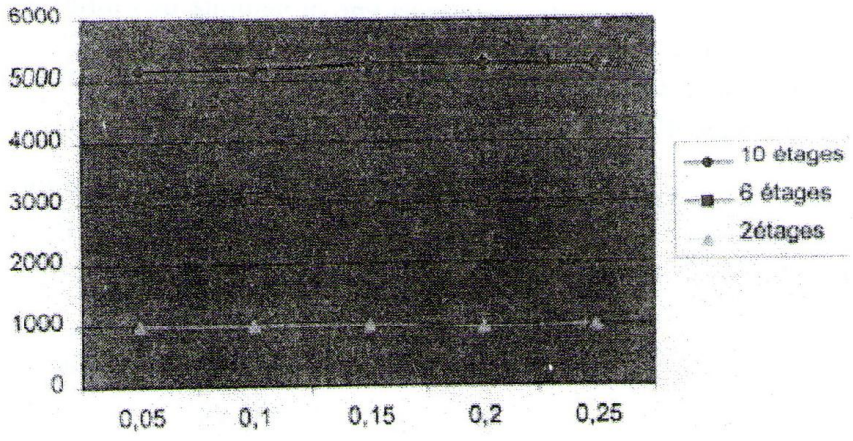


Figure 9 Influence de la probabilité de mutation.

La taille de la population est directement liée au temps de calcul utilisé par l'AG. Plus elle est grande et plus on pourra conserver la mémoire des bonnes solutions, tout en diversifiant la population et, donc, en l'améliorant globalement.

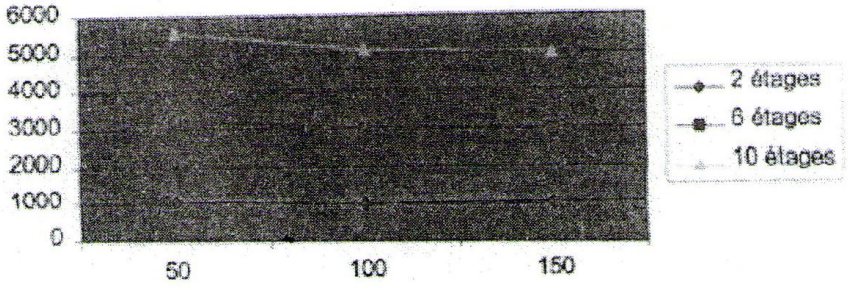


Figure 10: Influence de la taille de la population

IV.3 Améliorations de l'AG

Nos évaluations portent sur la version de base d'un AG. Ors, celle-ci propose de croiser ou muter des individus de la population et de les évaluer pour conserver le meilleur des individus rencontrés. En comparaison à un algorithme d'amélioration, même locale, la solution finale peut être mauvaise. Il est maintenant connu qu'il faut associer au principe de diversification de l'AG une technique d'amélioration (simple) de certains individus pour le rendre très efficace, même sur des problèmes très compliqués.

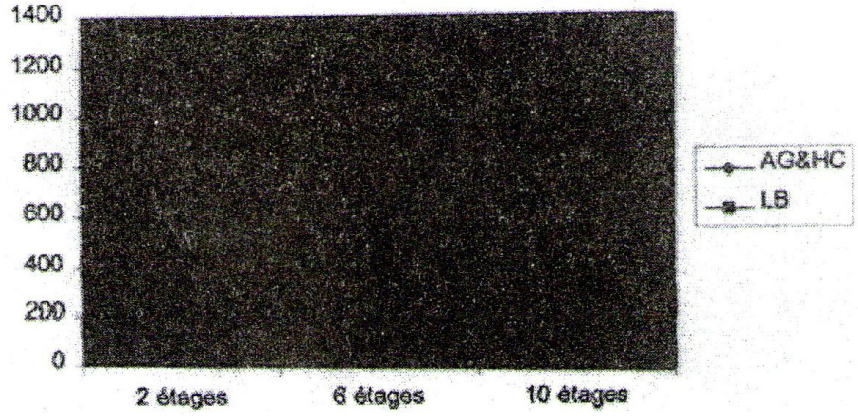


Figure 11 Comparaison de l'AG amélioré à la Borne inférieur de Santos

Nous avons mis en œuvre un AG combiné à une technique d'amélioration locale: Hill-Climbing. Quel que soit le jeu d'essai considéré, on note une amélioration des résultats au prix d'un surcoût de calcul significatif.

V Conclusion

Après avoir présenté l'algorithme génétique de base, nous l'avons appliqué au FH. Choisir des valeurs de paramètres de manière empirique sans vérifier leur adéquation à la résolution des FH aurait été trop rapide. Nous avons donc mené une campagne de tests pour valider les valeurs de ces paramètres. Ainsi nous donnons à notre AG les meilleures chances d'obtenir de bonnes performances.

Les résultats obtenus confirment les valeurs standards comme de bons choix: pcross=75%, pmut=15% et taille population=100. Les performances de l'AG restent cependant médiocres, en moyenne à 200% de la borne inférieure proposée par Santos [Santos et al.,95]. Nous avons donc modifié l'AG pour le combiner à une recherche d'un minimum local (Hill-Climbing). Cette combinaison rend l'AG très performant au prix d'un temps de calcul doublé.

Références

[Brah et al.,91] S. A. Brah et J.L. Hunsucker. *Branch and bound algorithm for the Flowshop with multiple processors*. European Journal of Operations Research, (51) : pp 88-99, 1991.

[Caux et al., 95] C. Caux, H. Pierreval et M-C. Portmann. *Les algorithmes génétiques et leur application aux problèmes d'ordonnancement*. APII vol. 29, n° 4-5, pp 409- 443, 1995.

[Holland,75] J. H. Holland. *Adaptation in natural and artificial systems*. Ann Arbor : University of Michigan Press, 1975.

[Gen et al.,97] M. Gen et R. Cheng. *Genetic Algorithms and Engineering Design*. W. H. Freemann, wiley edition, 1997. 411p.

[Djerid et al.,96] L. Djerid, M-C. Portmann et P. Villon. *Performance analysis of permutation cross-over genetic operators.* Journal of Decision Systems, 5(1-2) : pp 157-177, 1996.

[Portmann et al.,96] M-C. Portmann, A. Vignier, D. Dardilhac et D. Dezalay. *Some hybrid Flowshop scheduling by crossing branch and bound and genetic algorithms.* In Proceedings of 5th International Workshop on Project Management and Scheduling (PMS'96), Poznan (Pologne). Pp 186-189.

[Guinet et al.,96] A. Guinet, M.M. Solomon, P.K. Kedia et A. Dussauchoy. *A computational study of heuristics for two-stage flexible Flowshops.* Int. J. Pro. Res., 34(5) : pp 1399-1416, 1996.

[Goldberg,89] D.E. Goldberg. *Gentic Algorithms in Search, Optimization & Machine Learning.* Addison-wesley Publishing Company, Inc.

[Gupta,88] J. N. D. Gupta. *Two-stage hybrid Flowshop scheduling problem.* J. Op. Res. Soc., 39(4) : pp 359-364, 1988.

[Santos et al., 95] D. L. Santos, J. L. Hunsucker et D. E. Deal. *Global lower bounds for the Flowshop with multiple processors.* Eur. J. Op. Res., (80) : pp 112-120, 1995.

[Moursli,99] O. Moursli, Scheduling the Hybrid Flowshop : Branch and Bound Algorithms. Thèse de Doctorat de l'université Catholique de Louvain, 1999.

[Haouari et al.,97] M. Haouari et R. M'Hallah. *Heuristic algorithms for two-stage hybrid Flowshop problem.* Op. Res. Lett, (21) : pp 43-53, 1997.

[Lee et al.,94] C. Y. Lee et G.L. Viaraktarakis,. *Minimizing makespan in hybrid Flowshops.* Op. Res. Lett. (16) : pp 149-158, 1994.

[Vignier,97] A. Vignier. *Contribution à la résolution des problèmes d'ordonnancement de type monogamme, multimachine « Flowshop hybrid ».* Thèse préparé au LI de l'Universté de Tours, E3i. 1997.

[Nowicki et al.,98] E. Nowicki et Czeslaw Smutnicki. *The flow shop with parallel machines : A tabu search approach.* Eur. J. Op. Res. (106) : pp 226-253,1998.

[Deal et al.,91] D. E. Deal et J.L. Hunsucker. *The two-stage flow- shop problem with m machines at each stage.* Journal of Information and Optimization Sciences, (12) : pp 407-417, 1991.

[Davis,91] L. Davis. *Handbook of Genetic Algorithms.* New York : Van Nostrand Reinhold.1991.

[Soukhal et al.,99].A. Soukhal, P. Martineau et J-C. Billaut. *Présentation générale de trois algorithmes de voisinage appliqués au Flowshop hybride.* Rapport Interne, Laboratoire d'Informatique, E3i, Université François Rabelais, Tours.(à paraitre).

[GOTHA,93] GOTHA. *Les problèmes d'ordonnancement.* RAIRO-RO, vol.27, n°1, pp 77-150, 1993.

[Djerid,97] L. Djerid. Hybridation d'algorithmes génétiques et de méthodes classiques de recherche opérationnelle pour résoudre des problèmes d'ordonnancement. Thèse de Doctorat de l'Institut National Polytechnique de Lorraine, INP LORRAINE, 149 p, 1997.